CI-1001 - 2s2022 - Especificação do Trabalho

ENTREGA: 22/02/2023 até às 23:59 pelo Moodle. O código completo deve estar compactado em um arquivo .tar.gz.

Objetivo: implementar o jogo “de-burger”, no qual o personagem deve atender a pedidos de clientes, montando corretamente seus lanches.

A tela do jogo é um mapa estático contendo um cenário com estações fixas e um chapeiro movimentado pelo usuário:

```
|     é uma parede lateral
-     é uma parede superior ou inferior
@     é o ponto de entrega dos pedidos
&     é o chapeiro
o     é a lixeira
[H]   é a estação de hamburger
[p]   é a estação de pães (parte de baixo)
[P]   é a estação de pães (parte de cima)
[Q]   é a estação de fatias de queijo
[S]   é a estação de salada
[F]   é a estação de fritas
[R]   é a máquina de refrigerante
```

Um exemplo de mapa encontra-se a seguir:

```
#--------|@|--------#
| [R]                |
| [F]              o|
|       &            |
|[p] [H] [Q] [S] [P]|
#--------------------#
```

O jogo começa com uma fila de pedidos a serem atendidos na ordem em que chegaram (política FIFO). Um pedido consiste de:

1. Um cliente (número inteiro único e sequencial);
2. Uma refeição, que vem de uma enumeração com as seguintes opções:
    - X-Burger
    - X-Salada
    - Combo 1 (X-Burger + Fritas + Refrigerante)
    - Combo 2 (X-Salada + Fritas + Refrigerante)
    - Combo Vegetariano (Pão com queijo + Fritas + Refrigerante)
    - Combo Vegano (Salada + Fritas + Refrigerante)

Ao iniciar o jogo, o usuário deve entrar com uma quantidade de pedidos para criar a **fila de pedidos** (*default*: 10). Dada a quantidade, os pedidos devem ser gerados aleatoriamente de acordo com as opções de refeição disponíveis. Logo, uma fila com 4 pedidos pode ter o seguinte formato de exemplo:

```
[cliente 1 | Combo 1] -> [cliente 2 | X-Salada] -> [cliente 3 | Combo 1] -> [cliente 4 | X-Burger]
```

Os primeiros cinco pedidos devem aparecer na tela do jogo, ao lado do mapa, para que o chapeiro possa prepará-los. Ao entregar uma refeição no ponto correto (@), o cliente compara com a refeição que consta no primeiro pedido da fila (criar função `verificaPedido`) e, caso esteja correto, o pedido deve ser desenfileirado, ajustando-se os pedidos que aparecem na tela do jogo de acordo com a fila. Se o pedido entregue estiver errado, seja pela refeição não estar correta ou pela ordem da fila não ser obedecida, o cliente vai embora insatisfeito (desenfileira o pedido e o jogador perde 1 ponto). Se o jogador errar três entregas, o jogo é perdido.

Cada **refeição** consiste de uma pilha de ingredientes, os quais estão dispostos nas estações citadas mais acima e devem ser montados pelo usuário por meio da movimentação do chapeiro As refeições disponíveis são:

1. X-burger `[p][H][Q][P]`
2. X-salada `[p][H][S][P]`
3. Combo 1 `[p][H][Q][P][F][R]`
4. Combo 2 `[p][H][S][P][F][R]`
5. Vegetariano `[p][Q][P][F][R]`
6. Vegano `[S][F][R]`

O chapeiro deve usar as teclas ASDW como direcionais e, dado um pedido, montar a refeição “encostando” na estação adequada. Cada “encostão” adiciona (`PUSH`) UM ingrediente do tipo da estação no pedido. Cada encostão na lixeira remove a refeição inteira (`destroiRefeicao`). Após montagem da refeição, o chapeiro deve se dirigir até o ponto de entrega dos pedidos (@) para que o cliente saia da fila.

Condições de *game over*:

- Se três pedidos foram entregues errado, o jogador perde imediatamente.
- Se cinco refeições foram jogadas no lixo, o jogador perde imediatamente.
- Se os pedidos foram atendidos corretamente e pelo menos uma das duas condições acima não foram satisfeitas, o jogador ganha.

As implementações das estruturas de dados devem ser feitas por alocação dinâmica sempre que possível. O núcleo do jogo deve ser todo feito via bibliotecas separadas para FILA, PILHA e LISTAS. A PILHA deve ser implementada a partir de uma lista ligada simples. A FILA deve ser implementada a partir de uma lista duplamente encadeada. Quaisquer detalhes adicionais

não mencionados nesta especificação podem ser modificados pelos estudantes, **desde que justificado detalhadamente nos comentários dos códigos entregues e na apresentação**. Os trabalhos podem ser feitos por NO MÁXIMO duas pessoas.

- Até 20 pontos adicionais podem ser atribuídos, a critério do professor, para quem fizer melhorias na dinâmica do jogo. Por exemplo, movimentação via setas do teclado, mapa carregado como arquivo de entrada passado pelo usuário, chegada aleatória e automática de pedidos em tempo real, implementação de temporizador para cada pedido, clientes que saem da fila antes de pegar o pedido, gráficos, modo infinito (pedidos chegam constantemente), modo em fases (após terminar com sucesso uma fila de pedidos, o programa gera uma outra fila com menos tempo para atendimento de cada pedido, aumentando a dificuldade), etc.
- Se o trabalho tirar mais de 100 pontos, a nota excedente poderá ser utilizada para complementar a nota de trabalhos anteriores.